(1)

AFIT/GCS/ENG/92D-12

A SYNTHETIC ENVIRONMENT FOR SATELLITE MODELING AND SATELLITE ORBITAL MOTION

THESIS

David L. Pond, Capt, USAF

AFIT/GCS/ENG/92D-12

AFIT/GCS/ENG/92D-12

A SYNTHETIC ENVIRONMENT FOR SATELLITE MODELING AND SATELLITE ORBITAL MOTION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Systems

David L. Pond

Captain, USAF

December 1992

Approved for public release; distribution unlimited

Acknowledgments

I am indebted to several people for their assistance and cooperation with this project. In particular, I would like to thank Capt Rex Haddix, for his insight on working with several of the peripheral devices used in synthetic environments. I would like to thank Capt David Tisdale, for his assistance in working with the synthetic environment's hardware connections and operation. I also want to thank LtCol Martin Stytz for his guidance and understanding; he was helpful in ways I cannot count. I am also indebted to LtCol Phil Amburn, for his insight discussions in handling several problems.

I especially wish to thank my wife, Virginia, and my children, Christine and David, for their patience, understanding, and love during this effort.

David Lamarr Pond

Table of Contents

List of Figures

List of Tables

AFIT/GCS/ENG/92D-12

Abstract

Analysts tasked to study satellites moving in space must often rely on photographs to conduct their analysis. Often these photographs do not adequately represent how satellites move in space, their spatial relationships, or their interaction with other satellites. Photographs also fail to provide a three-dimensional representation of the satellites. What analysts need is a three-dimensional synthetic environment that depicts satellites as they actually move and interact in space. This thesis addresses the early development of the Satellite Modeler (SM), a system that allows analysts to study satellite motion and maneuvering of computer aided design (CAD) models of those satellites in a three-dimensional orbit. The system contains equipment that allows analysts to view the models either in a head mounted display, on a high resolution television monitor, or on a computer workstation monitor. Analysts can also video tape a session in the synthetic environment. A voice recognition system allows the analyst to give voice commands to the synthetic environment that modify model movement or his view of the models, as well as commands to control other devices in the environment. Using a positional tracker to monitor position and direction of view, the modeler allows the analyst to move around the models to obtain different perspectives of how the models move in space and interface with other models. The modeler also allows the analyst to script the movements of models to duplicate the movements of actual satellites in space.

This is the first year of a three-year effort at the Air Force Institute of Technology (AFIT) towards creating a synthetic environment whereby analysts can better understand satellite operations. The modeler will prove beneficial in both analyzing and teaching satellite motion and interfacing in outer space.

A SYNTHETIC ENVIRONMENT FOR SATELLITE MODELING AND SATELLITE ORBITAL MOTION

I. Introduction

1.1 Background

Synthetic environments, or virtual realities, are playing greater and greater roles in current scientific study. Synthetic environments are being used for such purposes as visualizing the surface of Mars (Hitchner, 1992) and studying airflow over and through virtual aircraft (Levit, 1992).

1.2 Problem Statement

Analysts are tasked to study satellites to determine their operational capabilities and how they move in space. To perform this analysis, the analysts create 3-D models of satellites from photographs taken of actual satellites, using a sophisticated computer aided design (CAD) package. Once the models are created, the analysts study multiple 2-D images of the model. Although the analysts can study any 2-D image, they cannot get an accurate indication of what the actual satellite looks like as a three dimensional object in space, how it moves in space, or what changes it undergoes when interfacing with other satellites in space. This drawback severely hinders the analysts' capabilities to determine operational capabilities of current satellites or future requirements of satellites. Analysts also send these 2-D images to other facilities to allow other analysts to study the satellites and to teach students about satellites. Students have the same difficulty understanding satellite capabilities from the 2-D images as the analysts. To resolve some of these drawbacks, analysts asked AFIT to research the possibility of creating a synthetic environment allowing them to study the models as if the analysts were in space watching the actual satellites in orbit in outer space.

1.3 Objectives

The ideal environment for analysts to conduct this analysis would be from a "god's eye view" in outer space. This environment would allow the analysts to study the satellites as the satellites actually moved in space. Unlike astronauts in space, the analysts would not be tied to any space station or shuttle craft. They would be free to move anywhere in outer space, even through objects, as if they had no physical presence about them.

The primary objective of this research was to create a synthetic environment that represents such an environment, that simulates satellites moving in outer space and allows analysts to move anywhere within the environment. A rapid prototype of the system would be built, using object oriented design (OOD) and object oriented programming (OOP), to develop a basic system analysts could use to define specific requirements.

This is the first year of a three-year effort at developing such an environment. This effort is sponsored by the Joint National Intelligence Defense Staff (JNIDS).

1.4 Requirements

There were several general requirements for the synthetic environment. Table 1 summarizes these requirements. The models representing satellites had to accurately depict the satellites as they appear in space. The actual satellites have hundreds of devices such as antennas and solar panels, as well as devices that allow one satellite to interface with another. The models created using the CAD package include polygons to represent all these devices as well as polygons to describe satellite interiors. Since for the analyst's purposes these models best represent the actual satellites, the synthetic environment had to use these models to represent the satellites in the synthetic environment. To see model features in full detail, the synthetic environment display device had to be at least VGA capability (640 x 480 pixels).

The synthetic environment had to allow the analysts to place any model at any position and in any orientation. The analysts had to have full command of the environment and dictate how the models moved and how they themselves moved in the environment. They had to feel as though they could "get their hands on" any model at any time and control it completely to better understand it, as if they could physically grasp the satellite in their hands and manipulate it to provide the best views of the satellites. They also had to be able to move anywhere in the environment to change their perspective of the environment. This included the possibility of actually entering a model to study its interior, or to view the environment from the model's viewpoint.

The synthetic environment also had to correctly model satellite movement in outer space, including moving models in near real time. Models could not travel through other objects or other models in the synthetic environment. They could not connect to other models in ways not allowed by the actual satellites. If two or more models simulated satellites docking together in space, the docked models had to move as a single model in the environment, just as the satellites would move as a single unit. If the environment allowed satellite models to move in ways not actually possible, analysts would get a false indication of how the actual satellites moved and operated in outer space. This would destroy the original intent of the synthetic environment, which is to aid analysts in studying satellites in outer space.

The synthetic environment had to allow analysts to remove from display any models not of immediate concern. If the environment contained several models, it could overwhelm the analyst or make it difficult to concentrate on a single model or interaction between just a few models.

To simulate an outer space environment, the synthetic environment had to provide different lighting effects. Satellites in space receive light from such sources as the sun, the moon, stars, or even from other satellites or spacecraft. If there were no light sources in outer space, the satellites would stay in darkness and analysts could not perform any analysis. Analysts needed the ability to illuminate objects in space at any time. They also needed the ability to illuminate any features on a model at any time. Therefore, the environment had to provide stationary as well as mobile lighting effects to represent sunlight, moonlight, a mobile spotlight, etc.

Another requirement of the synthetic environment was to allow analysts to record a session in the environment on video tape. These video tapes would accompany the 2-D images analysts were sending to other facilities and would improve the students' abilities to understand satellite modeling and orbital motion.

Models accurately depict actual satellites
Display environment in at least VGA mode (640 x 480 pixels)
Move models in any manner and to any position in the environment
Allow analyst to move anywhere within the environment
Allow analyst to jump into a model
Move models to duplicate the movements of actual satellites in space
Treat "docked" models as a single model
Remove from display any models not of immediate concern
Allow analyst to apply different lighting effects in the synthetic environment
Allow analyst to video tape any interactive session

Table 1. Summary of Requirements

1.5 Overview

The rest of this document describes the steps taken to create the synthetic environment. The next chapter describes a literature review of past research in creating synthetic environments, including research at AFIT. Chapter 3 details the design approach taken to create the Satellite Modeler, with a discussion on how the hardware and software work together to allow a user to enter and operate in the synthetic environment. Chapter 4 describes the results of this effort, and Chapter 5 summarizes the work and provides recommendations for future work in satellite modeling and satellite orbital motion.

1.6 Conclusion

This chapter discussed the current methods used by analysts to study satellites in space and the problems they face in using these methods. It described the analysts' requirements for a different method of analysis, described the ideal environment for analysis, and stated the requirements of that environment.

II. Literature Review

2.1 Introduction

This chapter discusses past and current literature on the topic of synthetic environments. It identifies several synthetic environment systems, including the equipment used as well as the problems users have had working with these systems.

2.2 Historical Review

There are several methods of creating a synthetic environment. For a user to feel they are actually immersed in a synthetic environment, the synthetic environment must create a different world in which the user can interact (Bryson, 1992-1: 1.2). If a user turns his head in the real environment, he would see different objects in his new field of view. The same must be true in a synthetic environment. To determine what the user would see in the synthetic environment, a system creating it must know the user's location and in which direction the user is looking at all times. In a participative synthetic environment, where the user interacts with the environment or causes the environment to change its behavior, the user must be able to communicate with the environment by issuing commands causing a change in it. In a non-immersive environment, the user is simply a bystander watching the environment, a non-participant.

This research began by studying past research efforts at AFIT on synthetic environments. Gerken created a synthetic environment, called the Battle Manager, consisting of a Silicon Graphics workstation, a low resolution (less than 320 x 200 pixels) HMD, a Polhemus 3Space Tracker system with two sensors, and a VPL DataGlove. One of the Polhemus sensors was attached to the HMD and allowed the system to determine the user's location in the environment. The other sensor was placed on the VPL DataGlove. The Battle Manager used the position and orientation values returned by the DataGlove's sensor to select objects in the environment to act upon. The only method of inputting commands to the Battle Manager was by making gestures with the VPL

DataGlove. Experiments with Gerken's system aided immensely in determining the equipment configuration for the Satellite Modeler.

Bryson (Bryson, 92-1) described equipment for two typical, general purpose virtual environment systems. Bryson did not state a specific purpose for each system, but maintained they contained different components necessary to create a virtual environment. The first, most significant system consisted of a computer workstation to execute the main software program, perform calculations and to drive the display, a head mounted display (HMD), with headphones and a microphone, a voice recognition system, and a DataGlove[1]. The HMD displayed the environment to the user and blocked his real environment. The headphones provided sound the user would hear if he was actually in the synthetic environment, while the microphone allowed the user to issue voice commands. The glove allowed the user to enter a small set of commands and point to objects in the environment. Bryson (Bryson, 1992-2) discussed several potential problems with glove input. The user often has trouble recognizing when the glove has accepted a gesture as a command. Also, the user must remember up to ten specific hand positions that represent commands to the glove hardware.

A second, much simpler system Bryson discussed consisted of a computer workstation and a boom display, similar to a "monitor on a stand" the user can look into and see the environment. The boom monitor sits on a movable arm mounted on a platform. Users can move the monitor a limited distance in any of six directions. The boom contains a tracker system that specifies the monitor's, and thus the viewer's, location in the environment. This system also contained a glove to provide user interaction with the environment.

Levit and Bryson (Levit, 1992) discussed a synthetic environment, called the virtual windtunnel, used to visualize simulated aerodynamic flows. Their article discussed

[1]The glove used by Bryson and most often described in research use is the VPL DataGlove, by VPL Research.

the lessons they learned in developing the virtual windtunnel. Their system's peripheral devices consisted of a boom display, VPL DataGlove, a Polhemus Tracker, mouse, and computer keyboard. Their system was designed so the user could quickly enter the environment by looking into the boom display, and exit quickly by simply pulling away from it. Statements by Levit indicate the boom display has a total field of view of about ninety degrees. In their application, the user usually concentrates attention on the center of view where resolution is highest. For this application, they felt a field of view of ninety degrees was adequate.

The virtual windtunnel uses the VPL DataGlove as a primary input device. Levit and Bryson used only a few gestures to add, delete, move, and manipulate groups of visualization tools. By using just a few significantly different gestures, users did not have to recalibrate the DataGlove every time it was used. Levit and Bryson were convinced representing command inputs through hand gestures would not lead to better control of visualization tools, but had not yet found a better way.

One important feature of the virtual windtunnel is it allowed a participative mode as well as a non-participative mode. In the participative mode, a user examines the environment through the boom display, seeing the virtual windtunnel (in a ninety degree field of view) move around them. In the non-participative mode, the virtual windtunnel is displayed on a standard workstation screen. This mode allowed a group of people to preview the environment.

Hitchner (Hitchner, 1992) described a virtual environment system, called Virtual Planetary Exploration (VPE), used to investigate concepts and strategies for designing planetary exploration workstations, based on virtual reality, with its initial application to visualize the surface of Mars. Hitchner's system consisted of a 3D graphics workstation, a HMD, a six degree-of-freedom (DOF) position tracker, and a six DOF joystick. Hitchner used his system to model digital terrain data sets of Mars, such as those transmitted from NASA's Viking orbiter satellites. Users of VPE could study low resolution images in near real time, however, the low resolution did not allow useful visualization of the data. Users could also view images at a high resolution (hundreds of thousands of polygons), however, the system was incapable of updating the display very quickly. Users failed to feel immersed in the environment, while others tended to get disoriented or even suffered motion sickness.

Hitchner stated that 3D graphics workstations priced under $300,000 could display 2,000 to 5,000 polygons per frame at a rate of 30 frames per second. In creating a synthetic environment for virtual planetary exploration, he allowed his system to render all models at their highest resolution and suffer the low update rates. He called this method the "look and wait" method. His studies showed many people adapted to this method fairly quickly, although he himself was not satisfied with it. Hitchner did not specify what his system's frame rate was, but it had to display millions of polygons, compared to the 100,000 polygons displayed by the Satellite Modeler.

Jacoby (Jacoby, 1992) discussed the problems he encountered in developing a general purpose virtual environment simulator. To be general purpose, Jacoby's system had to use many diverse pieces of equipment. It consisted of a host computer and two graphics systems. The system also used two different types of trackers, a Polhemus tracker with two sensors and an Ascension tracker with three sensors to determine positional values. A VPL DataGlove attached to a Macintosh personal computer allowed for gesture commands. Voice inputs were capable through VocaLink software running on an IBM PC connected to the host computer. Displays were available either through a HMD or a boom display.

Jacoby's system, although extremely flexible, has several drawbacks, as he points out. The HMD's display provides very poor resolution (320 x 200 pixels), and the boom display only offers a resolution of 400 x 400 pixels. Trying to configure such diverse hardware resulted in compatibility problems. For instance, the Ascension tracker system would not work with the host computer's interface cards, so Jacoby had to drop the

Ascension from use. Also, speech recognition was limited in use due to the difficulty in training and using the system.

2.3 Conclusion

This chapter discussed several different synthetic environment systems, their purposes, and problems faced in their current configurations. This information as well as information from past research at AFIT was instrumental in designing and assembling the Satellite Modeler, which is discussed in Chapter 3.

III. System Design and Implementation

3.1 Introduction

This chapter discusses the design philosophy, the hardware used in the modeler, and the software created and adapted to suit the modeler. It also discusses how each of the requirements of the modeler is satisfied, including the methods used to satisfy them.

3.2 System Design Approach

The design approach taken was to use object oriented design (OOD) and object oriented programming (OOP) techniques to rapidly implement a prototype. Each of the software components that controlled the synthetic environment's hardware devices were considered objects. By using OOD and OOP, a majority of the software developed could be reused in future projects. OOD and OOP also allowed a complex system to be broken down into simpler subsystems. Previous research efforts at AFIT (Gerken, 1991), (Brunderman, 1991), (Simpson, 1991) resulted in C++ classes to control several components for synthetic environments. Therefore, to allow the use of existing software, I selected C++ as the development language.

Using existing software allowed quick development of a system that could create a synthetic environment that allows analysts to view realistic models of satellites. The sooner analysts could interact with the synthetic environment, the sooner they could tell if the system was providing the analytic capabilities they needed and determine if other requirements existed. Another goal for this approach was to develop a flexible system. There was a strong possibility the analysts would not have access to all system equipment at one time. Therefore, the modeler was designed so the analysts could use it with any combination of equipment. They did not need the full complement of equipment described below to use the Satellite Modeler. I did not want to tie myself to a workstation that had a Polhemus 3Space Tracker attached.

The assembled components, both hardware and software, had to form a system that would display a synthetic environment representing highly complex satellites in outer space to the analyst. Analysts would seem to be in outer space, watching and studying satellites as they orbit. They would have no physical presence and therefore would be free to move anywhere in space, even into or through objects to enhance their ability to analyze the satellites in orbit.

The analysts could interact with the environment, pick a satellite out of space and move it in any manner desired to achieve a better perspective. A satellite would not be moving as in orbit, but would merely be sitting in space, ready for the analyst to pick the satellite out of space and study it up close. The analyst could rotate or move the satellite in any direction that would enhance the satellite's appearance.

The analyst could also be a non-participant in the environment. They could watch the satellites in outer space as they perform their assigned orbits. The analyst would not alter any of the satellites in space, but would be a bystander, observing the satellites' orbits. The analyst could move anywhere in the environment, possibly even jumping into a satellite, such as a space shuttle, and look at other satellites to determine what the satellites look like from the space shuttle's perspective. The analysts' actions would not alter the satellites' movements in any way.

The analysts could change the time of day in space, adding sunlight or moonlight as desired, or applying a direct spotlight to illuminate specific areas of the satellites. This would allow the analyst to "enter" outer space at any time of day or night and at any point in outer space, be it in the earth's orbit or another planet's.

The synthetic environment does not completely completely simulate the analyst's real environment. The synthetic environment does not give the analyst an indication of how large a model is or how far a model is from the analyst. A model in space may appear to be small because it actually is small, or it may be large but may be located far from the analyst. There are no reference points in the synthetic environment to give the analyst an indication of distance or size. Therefore, the synthetic environment does provide a textual display of how far a model is from the analyst's view.

The next sections describe the hardware and software components that create the synthetic environment and allow analysts to operate in the environment.

3.3 System Hardware

Figure 1 displays the modeler's current hardware configuration. At the heart of the Satellite Modeler (SM) is a Silicon Graphics (SG) Crimson workstation, with a single 50 MHz processor and 64 MB of main memory. The SG is responsible foı running the software to create and maintain the synthetic environment, and driving the display to the display devices. With 64 MB of main memory, up to 8 models can be in memory at one time when the modeler is executing.

Peripheral devices connect to the SG and pass data between the device and the SG via the SG's serial ports. These devices display the environment to the analyst, position him in it, and allow communication with the SG while he is immersed in the environment. An analyst can also use the modeler using the SG alone, without any of the peripheral devices, if desired.

To display the synthetic environment to the analyst and block the real environment, the analyst wears a head mounted display (HMD). The HMD provides the analyst with a three-dimensional display. The system has been used unsatisfactorily with a HMD with a resolution of less than 320 x 200 pixels.

To allow the analysts to record their interactive sessions, the modeler uses a Sony 3/4-inch tape recorder. The SG monitor cannot display images in national television standard (NTSC) mode, so a Sony television monitor allows non-participants to view the analyst's current session while he views the environment in the HMD. Viewers can watch the currenı session or previous sessions recorded on tape. The monitor connects to a second output port on the recorder.

To locate the user in the synthetic environment, the analyst wears a Polhemus 3Space Tracker sensor attached to the HMD. As configured for the Satellite Modeler, the Polhemus 3Space Tracker contains a single sensor and a source. It returns the mobile sensor's position and orientation, in reference to the Polhemus' stationary source, to the modeler. Position values are expressed as x, y, z coordinates, while orientation values are expressed as azimuth, elevation, and roll. By using the HMD with the sensor attached, the modeler can track the analyst's current position and orientation in the environment. The modeler determines where the analyst is looking (his reference point) by defining a sphere of constant radius, with the analyst centered at the sphere. The Polhemus reads the sensor's orientation values and returns them to the modeler. The modeler calculates where the analyst's direction of view, based on the azimuth, elevation and roll, would intersect the sphere. That intersection point is then the new reference point. The Polhemus system returns position and orientation values as long as the sensor is within 60 inches of the source. Outside this range, the position and orientation values are unreliable.

Analysts had to have a means to input commands into the modeler when they were immersed in the synthetic environment. They also had to control what was taking place in the environment. I evaluated two different methods of providing input to the synthetic environment: voice commands, using the Voice Navigator System (VNS) connected to a Macintosh computer, and VPL DataGlove gesture commands. The VNS is a relatively inexpensive method of adding voice capabilities to a synthetic environment. It connects to a Macintosh computer that translates voice commands into system commands, then sends the system commands to the SG. Both the VNS and the VPL DataGlove require training. The user has to train the VNS to recognize the way they pronounce certain words, while the VPL DataGlove requires the user to train the DataGlove system to recognize gestures. However, the Voice Navigator System seems much more reliable in accepting verbal commands than the DataGlove is in accepting gestures. The VNS also allows a larger set of commands. The VNS can reasonably handle up to 30 verbal commands, while the

DataGlove can only recognize up to 10 gestures. Bryson (Bryson, 1992-2) recommended only two or three gestures if you planned on using the DataGlove as an input device. The DataGlove seemed much too difficult to use for the casual user. For these reasons, I decided to use the VNS as a method of inputting commands to the modeler while immersed in the synthetic environment.

To make the modeler more flexible, I developed a complete set of commands analysts could input from the SG keyboard as well as voice commands. If the analysts did not have the full system configuration, they could still use the modeler with just the SG workstation. They did not need any of the peripheral devices if they did not have them or did not want to enter the synthetic environment.

3.4 System Software

The hardware's role in the system is to read the environment's current status, receive command input from the analyst, and display the environment based on its current status. The software must tie all the information together to determine what information to send to the display. Appendix A contains a description of each of the system software components.

The existing software I adapted for the modeler was written in C++ and included classes by several programmers. Brunderman (Brunderman, 1991) created a class to track the Polhemus 3Space Tracker sensor, read and send data across an RS-232 port, and read in and display models in a polygonal geometry (GEOM) format. Tisdale's class (Tisdale, 1992) issues commands to the Sony 3/4-inch tape recorder, and Gerken's class (Gerken, 1991) provides displays to a standard HMD. Haddix (Haddix, 1993) created a class to receive voice commands from an external computer (the Macintosh with the Voice Navigator System). These classes were all modified to satisfy the modeler's requirements

The HMD class places the modeler in NTSC mode and sends the display over one of the SG's serial ports, using the RS-232 port class, through the recorder and into the

HMD and television. The Polhemus class also uses the RS-232 class, constantly polling the Polhemus system for the current position and orientation of the sensor, then reads those values returned by the Polhemus. The voice class reads numerical values through a serial port, again using the RS-232 port class, and translates each value into a system command. The recorder class acts as a remote control for the recorder, sending the same commands to the recorder as those sent by its remote control. The analyst can issue commands to start, pause, or stop recording, or to rewind the tape. Since the system was designed to be flexible, the HMD, Polhemus, recorder and voice classes are all independent of one another.

All models displayed in a scene at one time are considered multiple parts of a single satellite. A single C++ class creates and controls a satellite. This class displays the satellite and allows the analyst to manipulate each of the parts of the satellite, including the ability to rotate or translate each part, join the parts into a single satellite, or eliminate the display of one or more parts. The class was designed initially as a data structure, with tools affecting the data structures being implemented as procedures acting upon the data structures.

The main software component of the modeler is a C++ program executing on the SG. This program creates instances of C++ classes to establish and manipulate several objects in the environment, including the HMD, Polhemus system, recorder, voice recognition system, and GEOM files. The Macintosh executes a C program that drives a multi-level menu system for the voice commands.

3.5 System Commands

To achieve the stated objectives of this research, the analysts had to have a comprehensive set of commands to manipulate the environment. These commands would allow the analysts to dictate how the environment behaved, such as how models moved to simulate satellites moving in outer space and what lighting effects the environment used.

Appendix E contains a list of system voice commands, while Appendix F lists keyboard and mouse commands.

As discussed in the section on System Design Approach, the system can be used in either of two ways, depending on whether the analyst wants to study the features of a single satellite or a satellite's orbit. To allow the analyst to simulate picking a satellite out of space to study it more closely, one group of commands can direct a model to rotate about its center of origin or translate the model along either the x, y, or z axis in the environment. Using these commands, the analysts can also place a model anywhere they desire in the synthetic environment. The analysts can choose to move their own viewpoint and travel around the model, just as if they were in floating outer space, or they can remain stationary and command the model to move or turn, as if they commanded the satellite in space. Using either method, the analysts get a true three dimensional feel for how the satellite looks in space.

If the environment contains several models and the analyst controlling the environment is not studying all of them, the analyst can order the modeler not to display some of the models. The analyst can turn on any hidden model's display at any time.

To allow the analyst to select whether they are in outer space during the day or at night, the modeler contains five light sources: directly above, below, to the left, and to the right of the origin, as well as a mobile spotlight. At system startup, the light sources directly above and below the origin provide light to the scene. At any time, the analyst can turn on or off any of the five lights, as if ordering one or more suns to appear in given locations, or carrying a huge spotlight to illuminate areas of interest.

If the analyst is using the modeler without the Polhemus system, they can still move their viewpoint and reference point through keyboard or voice commands. This capability adds to the flexibility of the system and allows the analyst to use the system without the full complement of equipment. These commands allow movement of either the viewpoint or reference point in any direction. With these commands, the analysts can change their vantage point and study the models from a different perspective, allowing them to move freely in the environment as the ideal environment would allow them to move in outer space.

At any time the modeler is executing, the analyst can request help on what system commands are available to him. A help menu will appear in the analyst's view listing the predominant keyboard and voice commands the system will accept at that time.

3.6 System Operation

This section describes how to use the modeler to create a synthetic environment that allows the analyst to simulate visiting and controlling outer space, watching and studying satellites in orbit. Appendix D identifies the requirements of the modeler and lists the system commands and the system operation that satisfies the requirements.

Analysts have to study detailed features of individual models to determine a single satellite's performance capabilities. They also have to analyze satellite characteristics and performance when they are in orbit and interfacing with other satellites. To allow both types of analysis, the analyst can use the modeler in either of two ways. In the first method, the synthetic environment represents outer space and the analyst is watching one or more satellites. The system allows the analyst to pick a single satellite out of space and examine it closely, moving it in the environment and possibly going inside the satellite to study it in-depth. In the second method, the system places the analyst in outer space among several satellites. The satellites are moving in orbit and the analyst is studying the satellites, moving around in space to get better perspectives of each satellite, dictating the lighting to apply to the scene. The analyst is free to move anywhere in outer space, including jumping into a satellite involved in a complex maneuver with other satellites.

The first method represents the analyst in outer space with one or more satellites, such as space stations in orbit. The analyst can select one of the models, turn it around, study it, and gain a better understanding of what the satellite looks like as a three goal of this stage is to allow the analysts to analyze the models in the synthetic environment as the models simulate the maneuvers actually performed by satellites in space. The analyst conducts analysis in this stage as if in space with the satellites as the satellites are performing their maneuvers. The analyst starts the modeler, issuing commands to include the Polhemus system, HMD, and possibly the recorder and television. The analyst also specifies a script exists for the models, attaches the Polhemus sensor to the HMD, and dons the HMD. They then start the main C++ program from the SG keyboard, specifying how the modeler should be configured during that session. The modeler reads in the database files and the script. It determines the analyst's position in the synthetic environment from the values returned by the Polhemus sensor, and places the models at the positions specified by the first frame of the script. The analyst can order the modeler to begin executing the script. The analyst can pause or stop execution of the script at any time. During the course of the script, the analyst can move to any position in the environment, including moving inside any of the models, with a voice command. The modeler automatically positions the analyst at the center of the model jumped into and changes their reference point to that of the object their model is docking with. The model jumped into would normally block the analyst's view of the other models, so the modeler will not display that model. The analyst will be able to go through the scenario as if they were a satellite docking with another satellite.

If two satellites in space have just completed a docking maneuver, they then become a single satellite. After the modeler finishes executing the script and the models are docked together, the analyst can give the command to join the two models and in effect become a single model. The modeler will calculate a new center point for the newly created model. Any command to transform the models treats the two models as one.

In both methods, the modeler displays the analyst's current position in the environment, their distance from each model displayed, and the current model number. This information is constantly displayed at the bottom of the display. The distance from dimensional object. Since the models have polygons describing their interiors, the analysts can enter any model and analyze its interior, just as if they were inside the actual satellite. This feature provides the analyst with the capability to study the model in three full dimensions, not just multiple two dimensions provided by the CAD models.

The second method involves running the SM in two stages. In the first stage, the analyst selects the models to place in the synthetic environment and creates a script specifying how models should move in the environment. This stage satisfies the requirement to move the models in the synthetic environment just as the satellites move in outer space. The analyst does not conduct analysis during this stage, but instead prepares the synthetic environment for the second stage. The goal of this stage is to allow analysts to create a script that moves models from initial starting points to finishing points using translations and rotations, saving those transformations to be repeated in the second stage. This stage saves the commands that move the models in the synthetic environment, freeing the analysts from having to manipulate the models in the second stage. The modeler allows the analyst to create the script interactively, by giving commands to rotate and translate each model to each position for each frame of the script. To begin creating a script, such as two satellites docking together, the analyst issues commands to place the models at their initial positions and orientations, then enters a command to begin a script. This saves the transformations that placed the models at their current positions and orientations in the environment. The analyst then moves the models to their next position in the script and saves that as well. By repeatedly moving and rotating the models, then saving their transformations, the analyst creates a script simulating the docking scenario from the satellites' initial positions to their docked positions.

In the second stage, the analyst enters the synthetic environment with the models, moving around the models to gain new perspectives as the models move as specified by the script. This stage is designed to satisfy the analyst's main requirements of being in a synthetic environment representing outer space with satellites maneuvering in space. The models provides the analyst with an indication of how large the models in view actually are. Without distance measurements, the analyst could not distinguish whether a model they were seeing was a large model far away or a small model up close. All commands to rotate, translate, or turn off display affect the current model. The current model number identifies what model is the current model.

3.7 Conclusion

This chapter discussed the design issues of this project and the way I integrated the hardware and software to develop the modeler. It described the features of the modeler, including the use of voice and keyboard commands, the modeler's flexibility in that it can be used on the SG alone or with other components. This chapter matched the analyst's requirements of the system to the commands and operations that satisfied those requirements. It also discussed two different methods of using the modeler to conduct model analysis. The next chapter describes the work to date and identifies problems encountered during the research.

IV. Results

4.1 Introduction

This chapter provides results of the work performed to date, including problems encountered and the analysts' use of the system to date.

4.2 Observations

Using the concept of rapid prototyping proved to be an effective method of defining the analysts' requirements. Using it, analysts could describe specific analysis tools that satisfied the general requirements initially identified. From the general requirement of models simulating actual satellite movement came the idea of saving and executing a script. From the need to view models from any viewpoint came the idea of jumping into a model during a script. However, it often made the program's software design issues cloudy at best. Some of the enhancements or modifications to features required a complete rewrite of the original feature's implementation to effectively carry out the modifications. For instance, to effectively implement the transformation of joined models, I had to rewrite the transformation of a single object. The modeler's software design was in a constant state of change. This was true for the driver and the satellite class, since these were the only classes affected by analysts' requirements.

Using OOD and OOP proved to be extremely helpful. Once a portion of the implementation was designed and programmed, I no longer had to worry about that portion. I was able to break the overall system design into smaller, less complex concepts that were substantially easier to design and program. After modifying the classes for the Polhemus, HMD, voice recognition, and recorder, I never had to change them. Changes in the analysts' requirements did not have an effect on the software controlling the hardware.

During the course of its development the Satellite Modeler was used by other researchers at AFIT for displaying models of Wojszynski's (Wojszynski, 1992) and

Tisdale's (Tisdale, 1992) radar cross section data, Haddix's (Haddix, 1993) battlefield components, and the interior of an AWACS cabin. At times during development, I relied on their inputs to add features more general in nature and not necessarily restricted to satellite modeling. These features included features such as rotating or translating all models currently displayed. These ideas improved the performance of the modeler without satisfying any specific analyst requirements.

Analysts' reaction to the Satellite Modeler has been positive. After viewing a script on the SG monitor showing a Kvant satellite model docking with the model of a Mir space station, analysts stated they felt they would use the system extensively even without the HMD. They also derived some interesting information concerning the Mir space station. As an example of the usefulness of the system, allow me to report one insight the analysts achieved by using it.

The Mir contains two long solar panels. When expanded, these solar panels are very long and wide, but very thin. While the modeler was rotating the Mir, the solar panels seemed to disappear from view. The analysts thought this phenomenon was a flaw in the program until I moved the viewpoint closer to the Mir. The solar panels returned to view when we moved within a certain distance of the Mir. The solar panels were simply too thin to see from a given distance. Since the modeler displays the models at the same size as the real satellites, the modeler revealed to the analysts at what distance the solar panels would seem to disappear from view in space.

Attempts to increase the frame update rate have proved disappointing. The analyst's requirements for seeing the models at their highest resolutions at all times eliminate the possibility of using progressive refinement, so the modeler reads the list of original polygons into an array in main memory. This array required an average of 6 MB of storage space per model. Gargantini (Gargantini, 1989) and Weng (Weng, 1987) described a method to store a pre-computed list of polygons visible from different viewpoints. By implementing this method, I created lists of visible polygons from the front, back, top, bottom, left, and right sides. Since some polygons are visible from several different viewpoints, many polygons were in more than one list. This in effect tripled the amount of storage space required for each model. The amount of storage space required to represent a model then grew from 6 MBytes to approximately 18 MBytes of storage space. The number of GEOM files that could reside in memory for an SG with 64 MBytes of main memory was reduced from 8 to only 3. Using this method, a file describing a satellite space station would occupy over 100 MBytes of storage space, and so the modeler could not display the space station. Therefore, I deemed this method unacceptable.

To try to use preprocessing to increase the frame update rate, I chose not to duplicate the lists of polygons. Instead, the list for each view contained pointers to the array of polygons in the polygons array. Of the six views, up to three views were possible at one time. Unfortunately, using this method to display a single view for a model, the SG had to follow pointers from the view's list to the polygon's list to the array of pointers. The modeler took just as long following the pointers to the polygons, then pointers to the points array as it took rendering the entire original array of polygons. There was no increase in the frame update rate; however, the lists of pointers to the polygons increased a model's storage space by almost 50%. I chose to forego this method and display all polygons, letting the SG determine the polygons to render.

The system was developed primarily using the SG workstation as a stand-alone system and not using any of the peripheral devices. However, since the system was designed to be flexible and allow several different configurations, it could be tested in several phases. The system was tested initially using only keyboard commands to ensure command logic was correct, that the results of issuing commands were consistent with the intent of the commands. Almost all peripheral devices were initially tested one at a time. After the system command set logic was tested, the voice commands were tested to ensure the voice commands achieved the same results as the comparable keyboard commands.

The Polhemus 3Space Tracker system was then tested to ensure the sensor returned the user's correct position and orientation. The recorder, HMD, and television monitor were tested together, in NTSC mode, to ensure the display appeared in both the television monitor and HMD and that the recorder actually recorded the session. After all peripheral devices were tested independently, they were all tested as a complete system.

A large problem encountered during development and testing was that other students were constantly changing the equipment configuration to test new equipment. At times, newer equipment would not work the same as older equipment. A code section that worked with older equipment would have to be reaccomplished to find the malfunction, or the old equipment would have to be reinstalled. In the latter part of the system development, the Polhemus system failed to provide analyst position and orientation values back to the SG. This problem was not resolved.

4.3 Conclusion

This chapter described the problems encountered during the Satellite Modeler's development, how they were resolved and how the modeler was received by analysts. It identified other uses of the modeler and how some of the design approaches improved development. The next chapter summarizes the work performed to date and suggests ideas for future work on the modeler.

V. Summary and Future Work

5.1 Introduction

This chapter summarizes the research and discusses possible future work on the Satellite Modeler. Work should be concentrated on three areas, as identified by the analysts. The first area is to increase the number of tools used to manipulate a model and to increase the display update rates. The second area is to identify components on the satellites, such as antennas and solar panels, and to allow analysts to manipulate these components independently of the satellite body. Last, the analysts want a global view of all satellites in space to determine satellite coverage and communications capabilities.

5.2 Summary

Development began by studying previous efforts at AFIT on synthetic environments, and selecting those components of past projects that I considered appropriate and adaptable for the modeler. This saved valuable research time and provided me with a basic system that was reliable and time-proven. I then developed a prototype system that allowed analysts to see what their models looked like in the modeler, without allowing the analysts to move the models. The analysts then provided inputs to add features to help them conduct their analysis when using the modeler. After adding the requested features, I repeated the process, and the modeler evolved from a skeleton system to one that allows the analysts to perform rudimentary analysis on the models they have created to date. This incremental prototyping method helped define the analysts' requirements and build a more comprehensive set of tools the analysts could use. It also helped define and bring out other requirements.

The Satellite Modeler presented here allows analysts to enter a synthetic environment representing space, analyzing sophisticated models of the satellites they are tasked to analyze. This modeler allows the analysts to study the models in three full dimensions instead of the two dimensions they were previously limited to by the CAD models, giving them an opportunity to fully understand the satellites that the models represent. It also allows the analysts to move the models as the satellites actually move in space, allowing the analysts the capability to perform analysis on models that have duplicated satellite orbits in space.

Analysts have stated they believe the Satellite Modeler will prove extremely helpful in the future, and may eventually evolve into a tool not only to analyze satellites, but to control them as well.

5.3 Recommendations

One possible way of increasing the display's update rate is to improve the program that converts CAD file format to GEOM file format. The current program creates lists of polygons that the SG renders flat shaded. Gouraud shaded polygons create a much more realistic display, and a display of fewer polygons that are Gouraud shaded would produce models with the same level of detail. Another possibility is to create the lists of polygons in order of position, so all adjacent polygons are listed in order. In this manner, all polygons could be rendered using a t-mesh approach. The t-mesh method uses points from the previous polygon to help define the current polygon. Still another method is to use texture mapping to provide highly detailed features.

The modeler currently allows the analyst to jump into a model during a script. When the analyst jumps into the model, the modeler places him at the center of the model. As a future enhancement, the analyst should be allowed to specify where to jump into a model. As an example, analysts would want to jump into the cockpit of the space shuttle, so they were looking out the shuttle's front windows. To do this, they would have to specify where the jump location was for each type of model.

When devices are attached to satellites, they can usually move a limited amount in different directions. For example, an antenna connected perpendicular to the Mir space station's body may be able to move up to 60 degrees in any direction, and a solar panel may unfold 180 degrees from its folded position. Analysts want to limit how far they can move these peripheral devices using the modeler, as well as range of motion for two models connected. To do this, they would need to specify each type of peripheral device's range of motion and what type of device each model represents. Each peripheral device would have a connectivity file, specifying its range of motion when connected to other devices.

The Kvant satellite can only dock with the Mir space station at certain locations. Currently, the analyst can move the Kvant so it appears to dock anywhere, and can even appear to enter the Mir. Analysts would like the modeler to allow connection between devices only at locations they specify and to detect collisions. An analyst would include this information with the range of motion in the connectivity file. Also, the Kvant must move in a certain predescribed manner when preparing to dock with the Mir. If actual satellite orbital dynamics dictate a model's movement, the analyst would not be able to move a model in a way contrary to the actual satellite's movement.

Analysts have also indicated a need for a "global coverage" view of the satellites in space. In this view, the analyst's viewpoint would be at a great distance from the earth. He would see a model of the earth in the synthetic environment with models of satellites orbiting the earth. Upon demand, the modeler would display the area each satellite covered on the earth as well as the communications and intervisibility capability between satellites in space. With this view, analysts could quickly determine if a specific area of the earth was adequately covered by satellites, or if the satellites had to be moved to provide the coverage. The view would also indicate if satellites were in position to allow communication between two locations.

Eventually, analysts would like to manipulate actual satellites in space using the modeler. By entering the synthetic environment, an analyst could grab satellite components, such as retracted solar panels, and expand them to their full positions simply by making a gesture or giving a verbal command. The analyst could change the direction antennas were pointing, expand or retract panels, force two satellites to dock or disconnect, or possibly even alter satellite orbits using the modeler.

5.4 Conclusion

This chapter discussed recommendations for enhancing the Satellite Modeler. They included ways to make the display more realistic by increasing the frame update rate, adding new tools to help analyze models, and restricting model movement so the models only move in ways the actual satellites would move in space. The final recommendations were to provide a global view of satellite coverage over the earth.

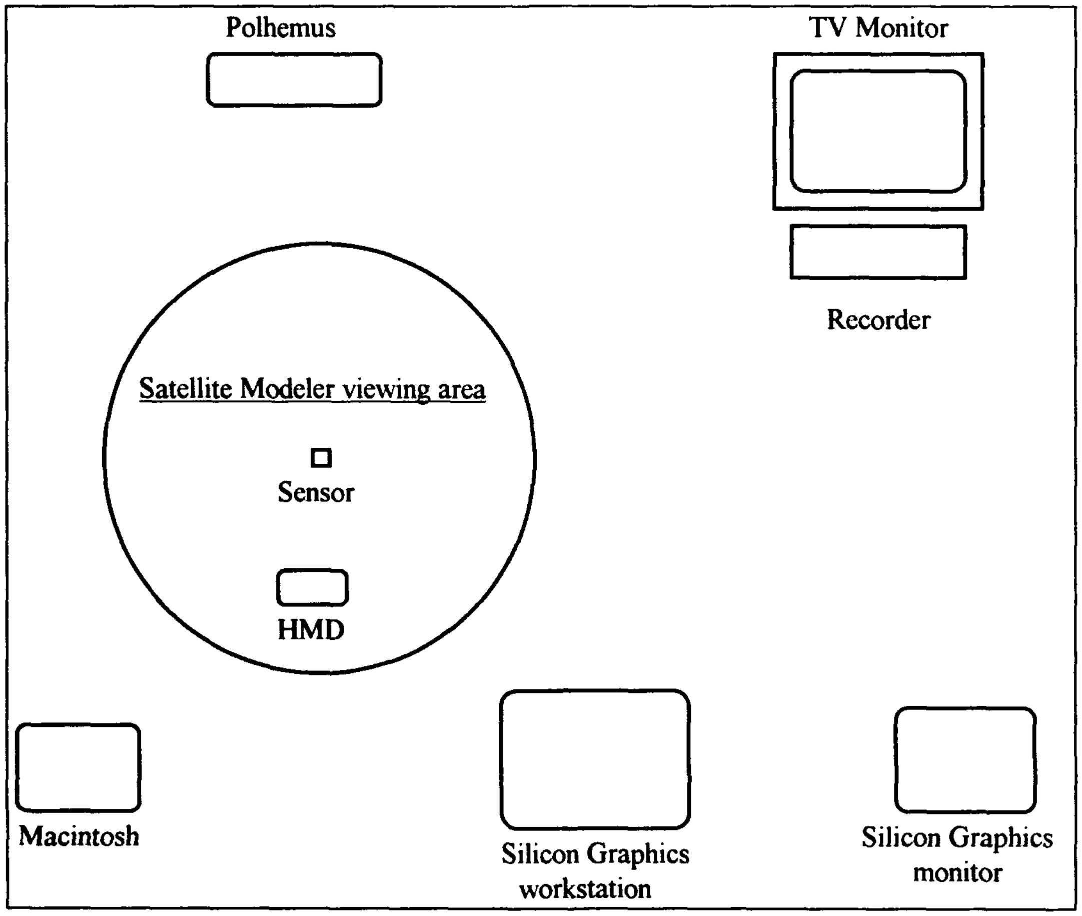

Figure 1. Satellite Modeler Configuration

Appendix A: System Design

The predominant work involved in this research was in creating or modifying C++ source code to control the modeler's hardware or to manipulate the models being displayed in the synthetic environment. Files were stored in the following directories:

Make files	-	/data5/dpond/thesis
Header files (.h suffix)	-	/data5/dpond/thesis/include
Source files (.c suffix)	-	/data5/dpond/thesis/src
Binary files	-	/data5/dpond/thesis/bin

File names and the functions they serve are as follows:

sm: The main executable program for the Satellite Modeler.

smobj.c: Source code for the main driver program. Reads in command line parameters to identify system configuration at start-up, receives inputs from Polhemus, voice system, keyboard, and mouse to determine what the synthetic environment will appear as. Describes the synthetic environment to either the monitor or HMD and issues commands to the recorder.

smobj.h: Procedure declarations for the main driver program.

objclass.c: Procedure definitions for the satellite object class. Defines the operations to perform on satellite models including procedures to read a database file (using the readfile class) identifying the models to display.

objclass.h: Variable declarations for the object class and procedure declarations.

geomclass.c: Procedure definitions for the GEOM file structure. Defines the operations to perform on a single GEOM model. Includes procedures to read a GEOM file from memory (using the readfile class) and display a GEOM.

geomclass.h: Variable declarations for the GEOM class and procedure declarations.

uRS232port.c: Procedure definitions to pass data between the SG, through its serial ports, to peripheral devices.

uRS232port.h: Variable declarations for the uRS232port class and procedure declarations.

readfile.c: Procedure definitions to read words and numbers in database and GEOM files.

readfile.h: Variable declarations for the readfile class and procedure declarations.

polhemus.c: Procedure definitions to poll and read the Polhemus 3Space Tracker for position and orientation information.

polhemus.h: Variable declarations for the polhemus class and procedure declarations.

voice.c: Procedure definitions to receive system commands from the Macintosh personal computer. The commands were input to the Macintosh via voice commands.

voice.h: Variable declarations for the voice class and procedure declarations.

recorder.c: Procedure definitions to command the video recorder to either record, pause, stop, or rewind.

recorder.h: Variable declarations for the recorder class and procedure declarations.

globals.h: All global variable and constant declarations used by all of the previous files.

In addition, the Macintosh contains C files to control the menu system, transmit data over the port to the Silicon Graphics workstation, and manage the windows on the Macintosh screen. These files are as follows:

voice.c: Controls the menu system on the Macintosh and translates the command received from the Voice Navigator System into a Satellite Modeler system command

voice.h: Header file for the voice.c.

voicep.c: Transmits the system command over the port to the SG.

voicep.h: Header file for voicep.c.

voicew.c: Controls the window management on the Macintosh display.

voicew.h: Header file for voicew.c.

Appendix B: Operations Manual

This section identifies how to use the Satellite Modeler, including command line parameter options to determine system configuration during the current session and commands to manipulate either the models in the synthetic environment or the user.

The Satellite Modeler is a C++ program called sm. There are several different ways to configure the modeler during your current session. You identify how the system is configured by entering command line parameters when starting the program. You start the program by entering

sm -d filename.ext [- options]

where filename.ext is the name of the database file listing all the models to display in the session. The format of the database file is as follows:

comment line describing what the file contains
objects n
file complete path name for the first model

.

.

.

file complete path name for the last model

The analyst must specify a database file to the system. A **-d** parameter indicates to the system that the database filename immediately follows. This is the only required parameter. A **-p** option specifies the analyst will be using the Polhemus 3-Space Tracker. During program execution, the analyst's viewpoint will depend on the sensor's position and his reference point will depend on the sensor's orientation. A **-n** option indicates the display is in NTSC mode. In normal mode, the system creates an 800 x 800 window in the computer monitor. The window does not take up the full screen. In NTSC mode, the system changes the window size to 640 x 480. The display will appear in a HMD and, if included, a television. The window will fill the entire viewing area in the HMD and on the television. The computer monitor will not provide any display. A **-l** option prepositions all models in the field of view. If models aren't prepositioned or placed by a script, the system centers all the models about the origin. When the analyst requests prepositioning, the system calculates each model's size when reading it into memory and places it in the field of view, equidistant from the other models. If the analyst doesn't provide a script, he should preposition the models with the -l option. A **-s** option specifies the database file contains a script. Each model in the database file will be transformed to the position and orientation specified in the first frame of the script. The system creates an array of transformation matrices, reads the script, and loads the script into the transformation matrices. A **-r** option indicates the system includes the 3/4-inch tape recorder. If the analyst types in just the system name (sm), or includes options **-h** or **-?**, the system displays a list of these options, then exits.

Not all combinations of options are possible. The system can only record when it is in NTSC mode, so the **-r** option should only be used when the **-n** option is also used. Also, if the analyst is wearing the HMD, he cannot move his viewpoint or reference point from the keyboard, so he should be wearing a Polhemus sensor. This means the analyst should specify both the **-n** option and the **-p** option. Finally, if the analyst indicates a script is part of the database file, he should not request the system preposition models. Therefore, the **-l** and **-s** options should not be combined. The script positions each model at the location specified by the first frame of the script.

After the user enters all options, the system reads the database file, then each of the GEOM files, and finally the script, if specified. The system calculates the size of each model, and if the user requested prepositioning, places the models around the origin. If the user indicated scripting, models are placed at the location and orientation specified by the first frame of the script. If neither scripting nor prepositioning is specified, the system centers all models at the origin, one model within another. After the system places all models, it moves the viewpoint back from the objects far enough so all objects are clearly in view. The larger the models are in the scene, the farther back the system places the viewpoint.

Creating a Script

A script is a predefined series of transformations applied to each model in the scene. The script allows the user to predefine all model placements and movements, simulating such actions as one or more satellite modules docking with a space station. To create a script, the user should start the system with the **-l** option (prepositioning the models), then move the models where desired at the start of the script. When the user indicates to the system they want to create a new script (by pressing the "c" key, for create), the system will take each model's current position and orientation as the first frame of the new script. The user will move each model to its next position and orientation, then press the "c" key again. The user can repeat this process and create up to 500 frames for a script. They end the script-creating process by pressing the "e" key (end create). Before exiting the system, the user must write the script to the database file by pressing the "w" key. This attaches the series of transformations to the end of the database file.

Executing a Script

After they have carefully created a script from detailed photographs of satellite modules in motion, users can view the models in motion, executing the script, while wearing the HMD and Polhemus sensor. This configuration will allow the user to move around the models while they interface, seeing the models from several different perspectives. The user gives the command to begin executing a script either by pressing the "s" key once or by giving a voice command of "Script On." The system will execute the script until either the user presses the "s" key a second time, gives a voice command of "Script Off," or the script executes to completion. The analyst can move their viewpoint, either by giving keyboard commands or changing position while wearing the Polhemus sensor, during the script. Once the script stops for any reason, the user can move any of the models.

Calling For Help

The system uses 32 different words to create 50 unique voice commands. There are 62 commands available via the keyboard. These commands are not obvious to the user. To help the user remember them, the system displays the most often used keyboard commands and their equivalent voice commands when the user requests help. The system displays this help menu either when the user presses the "h" key or gives the voice command "Help On." When the system includes the Voice Navigator System (VNS), the Macintosh computer will display a menu of all voice commands available at that time. The user turns off display of the help menu by hitting the "h" key a second time or giving the voice command "Help Off."

Selecting a Model

When the user wants one of the models to perform some action, such as rotating about an axis, he must indicate to the system which model they mean. A text display at the bottom of the window identifies which model is currently selected. The user selects a different model in one of two ways. They can either press the number key, along the top row of the alphabetic keyboard, identifying the model desired, or they can select the model by pressing the left mouse button. Each time the left mouse button is pressed, the system selects the model with the next higher number. After the system selects the last model (the one with the highest number), it selects the first one again. Each time the system selects a model, it displays it temporarily in red. This helps the user identify that model.

Moving the Viewpoint

Moving the viewpoint is one of the most important features of this system. The user must have ways of changing how they see the models in the scene. The user can change their viewpoint either by entering keyboard or mouse commands, voice commands, or changing their position while wearing the Polhemus sensor.

The number pad keys on the keyboard change the user's viewpoint. Number 2 lowers their viewpoint, number 8 raises it, 4 shifts it to the left, 6 shifts it to the right, number 1 shifts his viewpoint in and 0 shifts it out. Once they press any of these keys, the user's viewpoint continues shifting in that direction until they press the number 5 key, which stops shifting the viewpoint.

Voice commands that change the viewpoint are 'Viewpoint Left', 'Viewpoint Right,' etc. If the analyst is wearing the Polhemus sensor, they can change their viewpoint simply by moving position.

The user can also change their viewpoint during execution of a script by issuing a 'jump' command. When the jump command is issued, the user's viewpoint will change to the location of the currently selected model. The user will appear to be inside the model. Since the purpose of this command is to give the user a view from the model's perspective, the system does not display the model jumped into. Therefore the model does not obstruct the user's view. The user can issue the jump command either by pressing the 'j' key or by pressing the right mouse button. They can jump into a new model by selecting that model, as explained above. The user returns to their previous viewpoint by pressing the 'k' key or by pressing the middle mouse button.

Moving the Reference Point

Moving the reference point is as important as moving the viewpoint. The reference point determines what the user is looking at. As with the viewpoint, the system allows the user to change the reference point either by keyboard or mouse commands, voice commands, or changing the Polhemus sensor's orientation.

Function keys F3 through F8 translate the user's reference point either left, right, up, down, in, or out of the scene. Once they press any of these keys, the reference point continues moving until the user presses the F2 key. This stops the reference point's translations. Voice commands are similar to keyboard commands. These commands allow the analyst to translate the reference point in any of six directions.

If the user is wearing the Polhemus sensor, they can change the reference point by changing the orientation of the sensor. Another way to change the reference point is by jumping into one of the models. If the user is not using the Polhemus sensor, this will not only change the viewpoint, but will automatically change the reference point to that of the other model in the scene. The user can return their viewpoint and reference point to their position before jumping by pressing the 'k' key.

Transforming Models

Transforming a model involves changing either its position (translating) or its orientation (rotating). The system allows the user to move any model in a scene independently of any other model. To translate a model, the user selects which model they want to translate. They translate the model left either by pressing the 'Del' key or giving the voice command 'Move Left.' Models translate right by pressing the 'Page Down' key or giving the voice command 'Move Right.' They translate up by pressing the 'Home' key or giving the voice command 'Move Up.' They translate down by pressing the 'End' key or giving the voice command 'Move Down.' They translate towards the user by pressing the 'I' key or giving the voice command 'Move In.' They translate away from the user by pressing the 'O' key or giving the voice command 'Move Out.'

When the user gives a command to translate a model, the model continues translating from its original location by the same increment until the user gives the command to stop translating. The command to stop translating the current model is either the 'Enter' key or the voice command 'StopMoving.' If the user gives the command to move a model in one direction, such as left, then gives a command to move it in another direction, such as up, the model will combine the moves and move it left and up each frame. If they give commands to move a model in opposite directions, such as left and right, the system will consider the model to be translating, but the translated values will offset. The system will go through the process of translating the model twice each frame, so it will appear it won't be moving. The user should give the command to stop the model from translating at this point.

Rotating models is similar to translating models. The user gives the command to rotate a current model about the x axis by pressing the 'X' key or by giving the voice command 'Rotate XAxis'. To rotate about the y axis, the commands are issued by pressing the 'Y' key or giving the voice command 'Rotate YAxis', and about the z axis by pressing the 'Z' key or issuing the voice command 'Rotate ZAxis'. Models will continue to rotate about their center of origin along the specified axis until the user stops rotation by pressing the 'Backspace' key or issuing the voice command 'StopRotating'. Models can translate and rotate at the same time.

Displaying Models

The user can turn display of models on or off. At system start-up, all models are displayed. To turn off the currently selected model's display, the user presses the 'N' key. The 'V' key turns the model's display back on. Voice commands to turn display off are 'DisplayOff', and 'DisplayOn' to turn display back on.

Changing the Lighting

There are five light sources in the Satellite Modeler. These light sources are at the top, bottom, left, and right of the display as well as a mobile spotlight. At system start-up, the light sources at the top and bottom are turned on. To turn on or off any of the light sources, the user presses the 'L' key, then presses any of the arrow keys on the number pad. Number 2 selects the bottom light, number 8 the top light, number 4 the left light, and number 6 the right light. After selecting the light to turn on/off, the user presses either the '+' key on the number pad to turn a light source on, or the '-' key to turn it off. After turning the light source on or off, the user presses the number pad 'Enter' key to activate the switch.

To turn on or off the mobile spotlight, the user presses the 'L' key, then the 'S' key, to select the spotlight, then presses either the '-' or '+' keys followed by the 'Enter' key, as before. To move the spotlight, use any of the four arrow keys on the keyboard.

Text Displays

The screen always displays the distance from the user to the models onscreen. This gives the user an indication of how far away they far from the models as well as some idea of how large the models onscreen are. It also displays the currently selected model number.

Resetting the System

If the user decides they want to return the modeler to its original display at system start-up, they can do so by pressing the 'Space Bar' or by giving the voice command 'Reset'.

Exiting the System

The user exits the modeler by pressing the 'Esc' key or by giving the voice command 'ExitSystem'. This stops execution of the Satellite Modeler program.

Appendix C: Known Bugs

There are some cautions when using the Satellite Modeler. When some commands are issued at the wrong time, unpredictable results may occur. One example is when jumping into a model. This command was designed to be used when the modeler is executing a script. If the command is given when no script is being executed, the user may jump into a model, but their reference point may be unknown.

Another unsolved problem occurs when models are joined. If models are joined, and the user issues a command to translate the models, then issues a command to rotate them, the models will not rotate about their own center of origin, but the center of the rotation will be shifted by a small amount. I did not have the opportunity to resolve ths problem.

Finally, a warning about using the voice commands. There are four levels of menus for the voice commands. For some reason, the VNS only accepts commands to the third level. This only affects commands to torn light displays on or off, or to move the mobile spotlight. All other commands function as expected.

Appendix D: Requirements and Implementations

Requirements	How Implemented
Models accurately depict actual satellites	Display models created from CAD program
Display environment in at least VGA mode	Environment displayed on high resolution monitor or high resolution HMD
Allow analysts to move models in any manner and in any orientation in environment	Rotate and translate commands affect any model
Allow the analyst to move anywhere within the environment	Read the Polhemus sensor attached to the HMD to read the analyst's position and orientation, or change position or orientation with keyboard commands
Allow analyst to jump into a model	Jump into command during script execution
Move models to duplicate the movements of actual satellites in space	Second method of system operation: in first stage, define the actual movements of models to duplicate satellite movement, then analyze these movements in synthetic environment in second stage
Treat "docked" model as a single model	Unite command combines multiple models into one satellite
Allow analyst to apply different lighting effects in the synthetic environment	Four different stationary light sources and one mobile light source
Allow analyst to video tape any interactive session	Video recorder and commands to record, pause, and stop recording current session

Appendix E: System Voice Commands

Main Menu	Submenu								
Help --	On	Off	Cancel						
ExitSystem									
Rotate --	XAxis	YAxis	ZAxis	Cancel					
Move --		Left	Right	Up	Down	In	Out	Cancel	
Viewpoint --	Left	Right	Up	Down	In	Out	Cancel		
Reference Point --	Left	Right	Up	Down	In	Out	Cancel		
Light --	Left	Right	Top	Bottom	Cancel				
Spotlight --	On								
	Off								
	Move	--	Left	Right	Up	Down	In	Out	Cancel
	StopMoving								
	Cancel								
DisplayOn									
DisplayOff									
Record --	On	Off	Cancel						
StopMoving									
StopRotating									
Reset									
Script --		On	Off	Cancel					

Appendix F:
System Keyboard and Mouse Commands

Key	Command	Key	Command
ESC	Exit Program	G	---
F1	Rotate Model clockwise	H	Help On/Off
F2	Stop translating RP	J	Jump to Model
F3	Translate RP -Z	K	Jump out of Model
F4	Translate RP +Z	L	Lighting mode
F5	Translate RP -X	;	---
F6	Translate RP +X	'	---
F7	Translate RP -Y	ENTER	Stop Translating Model
F8	Translate RP +Y	Z	Rotate Model about Z axis
F9	Rotate all about X axis	X	Rotate Model about X axis
F10	Rotate all about Y axis	C	Create a Script frame
F11	Rotate all about Z axis	V	Make Model Viewable
F12	Rotate Model counterclockwise	B	---
PRT SC	---	N	Don't display Model
1	Select Model 1	/	---
2	Select Model 2	SPACE	Reset system to start-up
3	Select Model 3	INS	---
4	Select Model 4	HOME	Translate Model +Y
5	Select Model 5	PAGE UP	---
6	Select Model 6	DELETE	Translate Model -X
7	Select Model 7	END	Translate Model -Y
8	Select Model 8	PAGE DOWN	Translate Model +X
9	Select Model 9	UP ARROW	Move Spotlight +Y
0	---	LEFT ARROW	Move Spotlight -X
-	---	RIGHT ARROW	Move Spotlight +X
=	Select next Model	DOWN ARROW	Move Spotlight -Y
\	---	<u>Number Pad Keys</u>	
BKSPC	Stop Model rotating	/	---
Q	---	-	Move Spotlight -Z
W	Write the script to file	+	Move Spotlight +Z
E	End the Script	0	Move Viewpoint -Z
R	Record On/Off	1	Move Viewpoint +Z
T	---	2	Move Viewpoint -Y
Y	Rotate Model about Y axis	3	---
U	---	4	Move Viewpoint -X
I	Translate Model -Z	5	Stop moving Viewpoint
O	Translate Model +Z	6	Move Viewpoint +X
P	---	7	---
[	---	8	Move Viewpoint +Y
A	---	<u>Mouse Keys</u>	
S	Turn Script On/Off	LEFT	Select next Model
D	---	MIDDLE	Jump out of Model
		RIGHT	Jump into model

Bibliography

(Airey, 1990) Airey, John M., and others. "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments." *Computer Graphics Proceedings, SIGGRAPH 90.*

(Bryson, 1992-1) Bryson, Steve. "Survey of Virtual Environment Technologies and Techniques." *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92.*

(Bryson, 1992-2) Bryson, Steve. Speaking during course on *Implementation of Immersive Virtual Environments, SIGGRAPH 92.*

(Brooks, 1988) Brooks, Frederick P., Jr. "Grasping Reality Through Illusion - Interactive Graphics Serving Science." *Conference Proceedings, Special Issue of the SIGCHI Bulletin, 1988.*

(Brunderman, 1991) Brunderman, John. *Design and Application of an Object Oriented Graphical Database Management System for Synthetic Environments.* MS thesis, School of Engineering (AU), Air Force Institute of Technology, Wright-Patterson AFB OH, Dec 91.

(Funkhouser, 1992) Funkhouser, T. A., and others. "Management of Large Amounts of Data in Inte:active Building Walkthroughs." *1992 Symposium on Interactive 3D Graphics, special issue of Computer Graphics, pp. 11-20, SIGGRAPH, March 1992.*

(Gargantini, 1989) Gargantini, I., and others. "Adaptive Display of Linear Octrees." *Computer Graphics, Volume 13, SIGGRAPH 89.*

(Gerken, 1991) Gerken, Mark J. *An Event Driven State Based Interface for Synthetic Environments.* MS thesis, School of Engineering (AU), Wright-Patterson AFB OH, Dec 91.

(Haddix,. 1992) Haddix, Rex. *The Synthetic Battle Bridge System.* MS thesis, School of Engineering (AU), Wright-Patterson AFB OH , publication pending.

(Hitchner, 1992) Hitcher, Lewis E. "Virtual Planetary Exploration: A Very Large Virtual Environment." *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92.*

(Jacoby, 1992) Jacoby, R. H. "Using Virtual Menus in a Virtual Environment." *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92.*

(Levit, 1992) Levit, Creon and Bryson, Steve. "Lessons learned while implementing the virtual windtunnel project," *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92.*

(Simpson, 1991) Simpson, Dennis J. *An Application of the Object-Oriented Paradigm to a Flight Simulator.* MS thesis, AFIT/GCS/ENG/91D-22, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1991.

(Teller, 1991) Teller, Seth J., and Sequin, Carlo H. "Visibility Preprocessing For Interactive Walkthroughs." *Computer Graphics, Volume 25, SIGGRAPH 91.*

(Tisdale, 1992) Tisdale, David, *Methods for Viewing Radar Cross Sectional Data in 3D.* MS thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1992.

(Weng, 1987) Weng, Juyang and Ahuja, Narendra. "Octrees of Objects in Arbitrary Motion: Representation and Efficiency." *Computer Vision, Graphics, and Image Processing, Volume 39, 1987.*

(Wojszynski, 1992) Wojszynski, Thomas, *Scientific Visualization of 3D Radar Cross Sectional Data.* MS thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1992.

Vita

David Lamarr Pond was born on January 24, 1955, in Indianapolis, Indiana. At the age of five, he and his family moved to Los Angeles, California, where he attended Venice High School.

On February 15, 1975, at the age of twenty, David married Virginia Marie Daniels. Nine days later, he entered the United States Air Force as a morse systems operator. David remained enlisted in the Air Force until 1987, with assignments to Kelly AFB, Texas; Misawa AB, Japan; and Elemendorf AFB, Alaska.

In April, 1987, David was commissioned a second lieutenant under the Airmen's Education and Comissioning Program. After an initial tour of duty at Tinker AFB, Oklahoma, he was accepted to the Air Force Institute of Technology, where he worked towards his Master of Science degree in Computer Systems.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 1992	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A SYNTHETIC ENVIRONMENT FOR SATELLITE MODELING AND SATELLITE ORBITAL MOTION	5. FUNDING NUMBERS
6. AUTHOR(S) David L. Pond, Capt, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCS/ENG/92D-12
9. [illegible] AGENCY NAME(S) AND ADDRESS(ES) Joint National Intelligence Development Staff 4600 Silver Hill Road Washington, DC 20389	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. [illegible] STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Analysts tasked to study satellites moving in space must often rely on photographs to conduct their analysis. Often these photographs do not adequately represent how satellites move in space, their spatial relationships, or their interaction with other satellites. What analysts need is a three-dimensional synthetic environment that depicts satellites as they actually move and interact in space. This thesis addresses the early development of the Satellite Modeler (SM), a system that allows analysts to study satellite motion and maneuvering of computer aided design (CAD) models of those satellites in a three-dimensional orbit. The system contains equipment that allows analysts to view the models either in a head mounted display, on a high resolution monitor, or on a computer workstation monitor. Analysts can also video tape a session in the synthetic environment. A voice recognition system allows the analyst to give voice commands to the synthetic environment that modify model movement or his view of the models. The modeler also allows the analyst to script the movements of models to duplicate the movements of actual satellites in space.

14. SUBJECT TERMS Graphics, Scientific Visualization, Surface Rendering	15. NUMBER OF PAGES 53
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL